BML MUNJAL UNIVERSITY™
FROM HERE TO THE WORLD

MILK MANAGE-MENT SYSTEM

Name: Dyuti Dasmahapatra

Registration no: 210C2030142

Enrollment no: 210424

Batch: CSE 1 (2021)

Introduction:

The Milk Management System is designed to manage dairy, members, customers, milk collections from members, sales to the customer and plant and all the dairy related processes.The Dairy Management System provides rate card features to collection managers so they can collect milk of different fat with proper cost.

A simple case of milk procurement application is considered where milk is being collected through a number of milk societies established in the milk procurement area. Each society may supply CM (Cow Milk) or BM (Buffalo Milk) or both types of milk. Each society is being supervised by a staff of a dairy plant. One supervisor may supervise more than one society.

Entities:

In this example, there are **seven entities** namely:

1. MILK
2. FARMER
3. SOCIETY
4. CITY
5. TOWN
6. MILK-RECEIVED
7. BILLS

Structures of these relations are described as follows:

- **MILK**(Membership_ID,Date,Time,Type of Milk,Quantity,FAT%,SNF)
- **FARMER**(Membership_ID,NAME,ADDRESS)
- **SOCIETY**(Society_ID,Supervisor-Name, Number)
- **CITY**(Society_ID,City-Name)
- **TOWN**(Society_ID,Town-Name)
- **MILK-RECEIVED**(Society no.,Date,Time,Vehicle No.,Quantity of Milk,FAT%,SNF)
- **BILLS**(Membership ID,Farmer's Name,Payment Period,Amount,Quantity of Milk)

Tables:

MILK:

Attribute Name	Type Of Attribute
MembershipID	Primary Key
Date	Single(date)
Time	Single(time)
Type of Milk	Single(varchar)
Quantity	Single(int)
FAT%	Single(int)
SNF	Single(int)

FARMER:

Attribute Name	Type Of Attribute
MembershipID	Primary Key
Name	Single(varchar)
Address	Single(varchar)

←Foreign key to MILK and BILLS

SOCIETY:

Attribute Name	Type Of Attribute
SocietyID	Primary Key
SupervisorName	Single(varchar)
Number	Single(int)

CITY:

Attribute Name	Type Of Attribute
SocietyID	Primary Key
CityName	Single(varchar)

TOWN:

Attribute Name	Type Of Attribute
SocietyID	Primary Key
townName	Single(varchar)

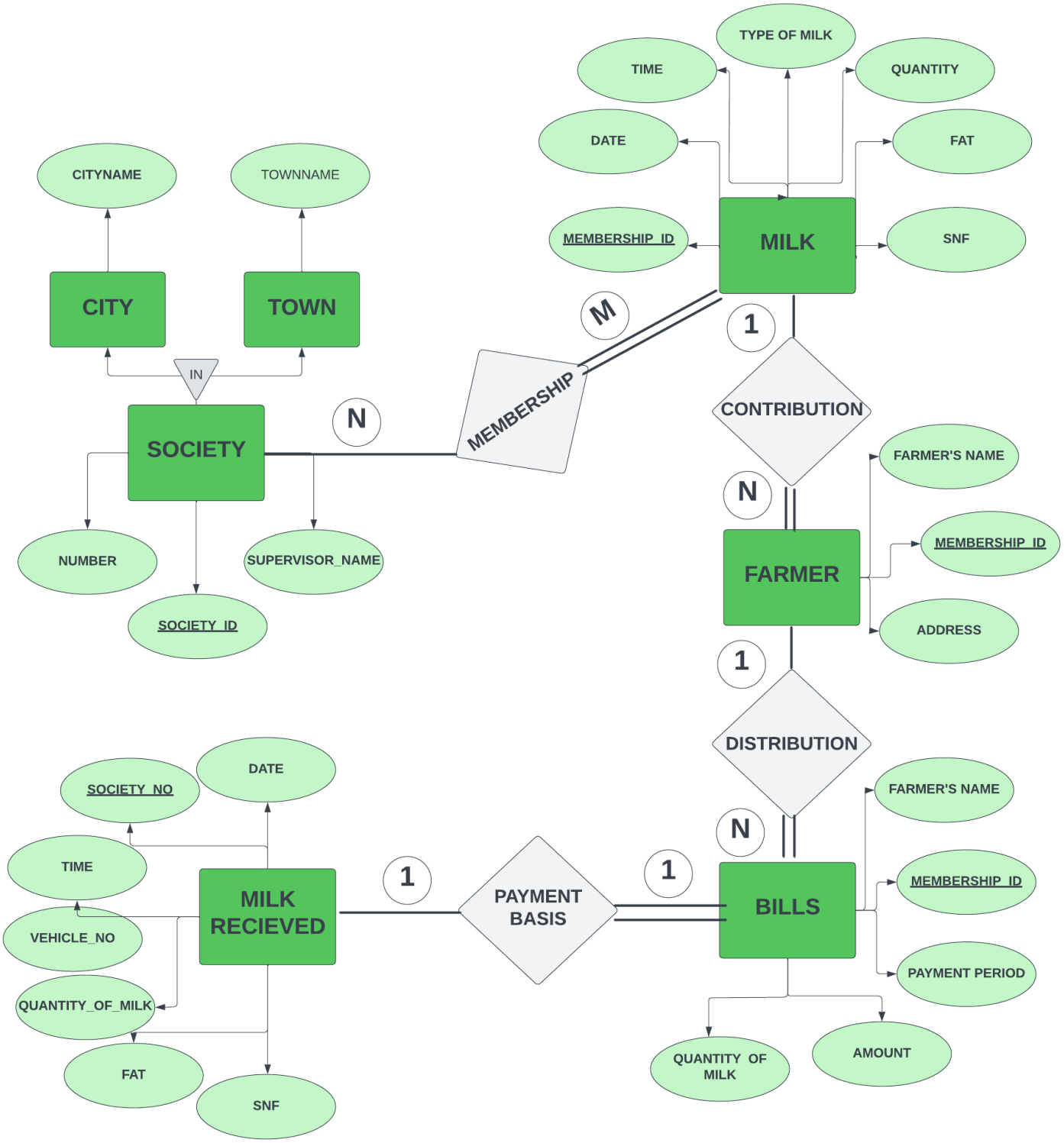
MILK-RECEIVED:

Attribute Name	Type Of Attribute
SocietyNo	Primary Key
Date	Single(date)
Time	Single(time)
VehicleNo	Single(varchar)
Quantity	Single(int)
FAT%	Single(int)
SNF	Single(int)

BILLS:

Attribute Name	Type Of Attribute
MembershipID	Primary Key
Farmer’s Name	Single(varchar)
Payment Period	Single(date)
Amount	Single(int)
Quantity of Milk	Single(int)

E-R Model:


TYPE OF MILK
TIME
QUANTITY
DATE
FAT
MILK
MEMBERSHIP_ID
SNF
CITYNAME
TOWNNAME
CITY
TOWN
IN
M
1
MEMBERSHIP
CONTRIBUTION
SOCIETY
N
N
FARMER'S NAME
MEMBERSHIP_ID
FARMER
ADDRESS
NUMBER
SUPERVISOR_NAME
SOCIETY_ID
1
DISTRIBUTION
DATE
SOCIETY_NO
FARMER'S NAME
N
TIME
1
1
MILK RECIEVED
PAYMENT BASIS
BILLS
MEMBERSHIP_ID
VEHICLE_NO
PAYMENT PERIOD
QUANTITY_OF_MILK
FAT
QUANTITY OF MILK
AMOUNT
SNF

Relationships:

Membership → MILK - SOCIETY n:m

as one milk service can distribute in many societies and one society can have more than one type of milk service.

Contribution → FARMER - MILK 1:n

as one farmer can produce more than one milk but one milk can be produced by one farmer

Distribution →BILLS - FARMER n:1

as there can be only one bill associated with one farmer but one farmer can have multiple bills associated with him.

Payment Basis → MILK_RECEIVED - BILLS 1:1

as only one milk can be sold with one bill and one bill can be associated with one milk sold

Normalization:

First Normal Form –

If a relation contains a composite or multi-valued attribute, it violates the first normal form. As the attributes in the taken example are all single valued attributes, the database is in 1st Normal form.

Second Normal Form –

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. In this database, it already is in its first normal form and does not have partial dependency.

Third Normal Form –

A relation is in 3NF if at least one of the following condition holds in every non-trivial functional dependency X –> Y
X is a super key and Y is a prime attribute (each element of Y is part of some candidate key).

In the "MILK" table given below, MembershipID→ Type of Milk, Type of Milk → FAT%, Type of Milk→SNF and Type of Milk is not a super key.

Attribute Name	Type Of Attribute
MembershipID	Primary Key
Date	Simple(date)
Time	Simple(time)
Type of Milk	Simple(varchar)
Quantity	Simple(int)
FAT%	Simple(int)
SNF	Simple(int)

To convert the above relation to 3NF, we need to split the table into two tables:

Table 1: MembershipID, Date, Time, Type Of Milk, Quantity

Table 2: Type of Milk, FAT%, SNF

TABLE 1

Attribute Name	Type Of Attribute
MembershipID	Primary Key
Date	Simple(date)
Time	Simple(time)
Type of Milk	Simple(varchar)
Quantity	Simple(int)

TABLE 2

Attribute Name	Type Of Attribute
Type of Milk	Primary Key
FAT%	Single(int)
SNF	Single(int)

Boyce-Codd Normal Form –

A relation is in BCNF iff in every non-trivial functional dependency X –> Y, X is a super key. The database satisfies the condition.

Sql Implementation:

1. Create a database named “milk_manage_system”

```
create database milk_manage_system; ## CREATE DATABASE NAMED "milk_manage_system"
use milk_manage_system ; ## To select a particular database to work with
```

2. Create all the entities as tables, with all attributes and primary keys defined.

```
create table MILK(Membership_ID int not null,Date date,Time time,Type_of_Milk varchar(40),
Quantity int, primary key(Membership_ID));
create table N_MILK(Membership_ID int not null ,FAT int,SNF int, primary key (Membership_ID) );
create table FARMER(Membership_ID int not null,name varchar(45),ADDRESS varchar(45), primary key(Membership_ID));
create table SOCIETY(Society_ID int not null,Supervisor_Name varchar(45), Number int,primary key(Society_ID));
create table CITY(Society_ID int not null,City_Name varchar(20),primary key(Society_ID));
create table TOWN(Society_ID int not null,Town_Name varchar(20),primary key(Society_ID));
create table MILK_RECEIVED(Society_no int not null,Date date,Time time,Vehicle_No varchar(10),
Quantity_of_Milk int,FAT int,SNF int, primary key(Society_no));
create table BILLS(Membership_ID int not null, Farmer_name varchar(30),Payment_Period date,
 Amount int,Quantity_of_Milk int, primary key(Membership_ID));
```

3. Add foreign key

```
ALTER TABLE MILK ADD FOREIGN KEY (Membership_ID) REFERENCES FARMER (Membership_ID);
ALTER TABLE BILLS ADD FOREIGN KEY (Membership_ID) REFERENCES FARMER (Membership_ID);
```

4. Insert values into the tables.

```
insert into BILLS(Membership_ID,Farmer_Name,Payment_Period,Amount,Quantity_of_Milk) values
 (1001,"Ram",20221112,200,4);
 insert into BILLS(Membership_ID,Farmer_Name,Payment_Period,Amount,Quantity_of_Milk) values
 (1002,"Suraj",20221113,100,2);
 insert into BILLS(Membership_ID,Farmer_Name,Payment_Period,Amount,Quantity_of_Milk) values
 (1003,"Akhil",20221112,50,1.5);
 insert into BILLS(Membership_ID,Farmer_Name,Payment_Period,Amount,Quantity_of_Milk) values
 (1004,"Mohit",20221114,150,3);
 insert into BILLS(Membership_ID,Farmer_Name,Payment_Period,Amount,Quantity_of_Milk) values
 (1005,"Naitik",20221114,20,1);
```

The select statement returns the records in the table.

Membership_ID	Farmers_Name	Payment_Period	Amount	Quantity_of_Milk
1001	Anvika	2022-11-12	200	4
1002	Rohit	2022-11-13	100	2
1003	Akshay	2022-11-12	50	2
1004	Priya	2022-11-14	150	3
1005	Ram	2022-11-14	20	1

Repeat the process to insert values into all the tables created.

Q.1 Display the name of farmers who produce buffalo milk.

We are going to use the **'Type_of_Milk'** column from **MILK** and **'name'** from **FARMER**.

```
select * from MILK;
```

Membership_ID	Date	Time	Type_of_Milk	Quantity
1001	2022-10-12	12:00:00	BM	20
1002	2022-10-12	11:30:00	CM	40
1003	2022-10-13	10:45:00	BM	30
1004	2022-10-13	12:25:00	CM	30
1005	2022-10-13	12:30:00	CM	40

```
select * from FARMER;
```

Membership_ID	name	ADDRESS
1001	Ram	Malana
1002	Suraj	Village
1003	Akhil	Majuli
1004	Mohit	Poovar
1005	Naitik	Kasol

The **INNER JOIN** keyword is used to select records that have matching values in both tables. The following command will display the name of farmers who produce buffalo milk

```
select * from MILK;
SELECT MILK.Type_of_Milk,FARMER.NAME
FROM MILK
INNER JOIN FARMER
ON MILK.Membership_ID = FARMER.Membership_ID
where Type_of_Milk="BM" ;
```

Type_of_Milk	NAME
BM	Ram
BM	Akhil

Q.2 Display the name of the society (city) where Ram supervises.

We are going to use "**Supervisor_Name"** from **SOCIETY** and "**City_Name** " from **CITY.**

```
select * from SOCIETY;
```

Society_ID	Supervisor_Name	Number
2140	Ram	999999999
2141	Suraj	888888888
2142	Akhil	777777777
2143	Mohit	666666666
2144	Naitik	666666666

```
select * from CITY;
```

Society_ID	City_Name
2140	Ispatika Apt
2141	Aashirvad Apt
2142	Salarpuri Apt
2143	Asha Apt
2144	Vatika Apt

Natural Join keyword joins two tables based on the same attribute name and datatypes. The following command will display the name of the society in the city, where Ram is the supervisor.

```
SELECT *
FROM SOCIETY
NATURAL JOIN CITY
where Supervisor_Name="Ram";
```

Society_ID	Supervisor_Name	Number	City_Name
2140	Ram	999999999	Ispatika Apt

Q.3 Display vehicle No. of vehicles which delivered milk on 13/10/2022.

From the database **MILK_RECEIVED**, "**Vehicle_No**" is going to be used

```
select * from MILK_RECEIVED;
```

Society_no	Date	Time	Vehicle_No	Quantity_of_Milk	FAT	SNF
2140	2022-11-12	15:23:00	HR47AF1371	20	1	2
2141	2022-11-13	13:23:00	HR47AF4253	20	3	4
2142	2022-11-13	14:23:00	HR47AF4567	20	3	2
2143	2022-11-13	15:33:00	HR47AF2153	20	1	2
2144	2022-11-14	15:40:00	HR47AF7456	20	1	4

The following command consists of a **SELECT** which will select and display the records, and a **clause WHERE** with the **condition** that Date is 2022-11-13.

```
select Vehicle_No from MILK_RECEIVED where Date=20221113;
```

Vehicle_No
HR47AF4253
HR47AF4567
HR47AF2153

Q.4 Display records of database BILLS, where the quantity of milk is greater than 1 and the amount sorted in decreasing order.

From database **BILLS,** all the columns will be used.

```
select * from BILL;
```

Membership_ID	Farmer_name	Payment_Period	Amount	Quantity_of_Milk
1001	Ram	2022-11-12	200	4
1002	Suraj	2022-11-13	100	2
1003	Akhil	2022-11-12	50	2
1004	Mohit	2022-11-14	150	3
1005	Naitik	2022-11-14	20	1

The following command consists of **SELECT** which will select records and display them, **two clauses WHERE and ORDER BY**, with the **condition** that Quantity_of_Milk is greater than 1 and the amount is in decreasing order respectively.

```
select * from BILL where Quantity_of_Milk>1 order by Amount desc  ;
```

Membership_ID	Farmer_name	Payment_Period	Amount	Quantity_of_Milk
1001	Ram	2022-11-12	200	4
1004	Mohit	2022-11-14	150	3
1002	Suraj	2022-11-13	100	2
1003	Akhil	2022-11-12	50	2